C interfaces to GALAHAD ROOTS

Jari Fowkes and Nick Gould
STFC Rutherford Appleton Laboratory
Thu Jun 22 2023

1 GALAHAD C package roots **1**

1.1 Introduction . . . 1

1.1.1 Purpose . . . 1

1.1.2 Authors . . . 1

1.1.3 Originally released . . . 1

1.1.4 Method . . . 1

2 File Index **3**

2.1 File List . . . 3

3 File Documentation **5**

3.1 galahad_roots.h File Reference . . . 5

3.1.1 Data Structure Documentation . . . 5

3.1.1.1 struct roots_control_type . . . 5

3.1.1.2 struct roots_inform_type . . . 6

3.1.2 Function Documentation . . . 6

3.1.2.1 roots_initialize() . . . 6

3.1.2.2 roots_information() . . . 7

3.1.2.3 roots_terminate() . . . 7

Chapter 1

GALAHAD C package roots

1.1 Introduction

1.1.1 Purpose

Use classical formulae together with Newton's method to find all the real roots of a real polynomial.

Currently, only the control and inform parameters are exposed; these are provided and used by other GALAHAD packages with C interfaces.

1.1.2 Authors

N. I. M. Gould, STFC-Rutherford Appleton Laboratory, England.

C interface, additionally J. Fowkes, STFC-Rutherford Appleton Laboratory.

Julia interface, additionally A. Montoison and D. Orban, Polytechnique Montréal.

1.1.3 Originally released

April 2005, C interface January 2022.

1.1.4 Method

Littlewood and Ferrari's algorithms are used to find estimates of the real roots of cubic and quartic polynomials, respectively; a stabilized version of the well-known formula is used in the quadratic case. Newton's method is used to further refine the computed roots if necessary. Madsen and Reid's method is used for polynomials whose degree exceeds four.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

galahad_roots.h 5

Chapter 3

File Documentation

3.1 galahad_roots.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "galahad_precision.h"
#include "galahad_cfunctions.h"
```

Data Structures

- struct roots_control_type
- struct roots_inform_type

Functions

- void roots_initialize (void **data, struct roots_control_type *control, int *status)
- void roots_information (void **data, struct roots_inform_type *inform, int *status)
- void roots_terminate (void **data, struct roots_control_type *control, struct roots_inform_type *inform)

3.1.1 Data Structure Documentation

3.1.1.1 struct roots_control_type

control derived type as a C struct

Data Fields

bool	f_indexing	use C or Fortran sparse matrix indexing
int	error	error and warning diagnostics occur on stream error
int	out	general output occurs on stream out
int	print_level	the level of output required is specified by print_level
real_wp_	tol	the required accuracy of the roots

Data Fields

real_wp_	zero_coef	any coefficient smaller in absolute value than zero_coef will be regarde to be zero
real_wp_	zero_f	any value of the polynomial smaller in absolute value than zero_f will be regarded as giving a root
bool	space_critical	if .space_critical true, every effort will be made to use as little space as possible. This may result in longer computation time
bool	deallocate_error_fatal	if .deallocate_error_fatal is true, any array/pointer deallocation error will terminate execution. Otherwise, computation will continue
char	prefix[31]	all output lines will be prefixed by .prefix(2:LEN(TRIM(.prefix))-1) where .prefix contains the required string enclosed in quotes, e.g. "string" or 'string'

3.1.1.2 struct roots_inform_type

inform derived type as a C struct

Data Fields

int	status	return status. Possible values are: • 0 the call was successful. • -1. An allocation error occurred. A message indicating the offending array is written on unit control.error, and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -2. A deallocation error occurred. A message indicating the offending array is written on unit control.error and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -3. Either the specified degree of the polynomial in degree is less than 0, or the declared dimension of the array roots is smaller than the specified degree.
int	alloc_status	the status of the last attempted allocation/deallocation
char	bad_alloc[81]	the name of the array for which an allocation/deallocation error occurred

3.1.2 Function Documentation

3.1.2.1 roots_initialize()

```
void roots_initialize (
             void ** data,
             struct roots_control_type * control,
             int * status )
```

Set default control values and initialize private data

Parameters

in,out	*data*	holds private internal data
out	*control*	is a struct containing control information (see roots_control_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The initialization was succesful.

3.1.2.2 roots_information()

```
void roots_information (
            void ** data,
            struct roots_inform_type * inform,
            int * status )
```

Provides output information

Parameters

in,out	*data*	holds private internal data
out	*inform*	is a struct containing output information (see roots_inform_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The values were recorded succesfully

3.1.2.3 roots_terminate()

```
void roots_terminate (
            void ** data,
            struct roots_control_type * control,
            struct roots_inform_type * inform )
```

Deallocate all internal private storage

Parameters

in,out	*data*	holds private internal data
out	*control*	is a struct containing control information (see roots_control_type)
out	*inform*	is a struct containing output information (see roots_inform_type)